Listing of Claims:

1. (Currently Amended) A method comprising:

at a computer, dynamically binding an event context to an execution context in response to receiving events by:

maintaining the execution context in an idle state until an event arrives at a head of a global event queue that is accessible by event contexts;

storing, in a memory associated with the computer, arriving events into [[a]] the global event queue ~~that is accessible by event contexts~~;

storing, in the memory, events from the global event queue in per-execution context event queues; and

associating, at the computer, an event queue with the execution context to temporarily store events for the event context for a duration of the dynamic binding.

2. (Previously Presented) The method of claim 1 wherein the execution context can be in one of four states, idle, binding, bound, or unbinding.

3. (Original) The method of claim 1 wherein in the bound state, an execution context is bound to a specific event context and the execution context processes events for that event context and the event queue associated with that execution context is used to store events for the event context to which it is bound.

4. (Original) The method of claim 1 wherein in the unbinding state, the execution context determines if it has any more events to process for the event context to which it was bound and either unbinds itself from the event context, going to idle state or begins processing another event from that context, going back to bound state.

5. (Original) The method of claim 1 wherein in the event context can be in one of two states, unbound or bound.

6. (Previously Presented) The method of claim 1 wherein a global FIFO event queue is used to queue events when the events first arrive into a system.

7. (Canceled)

8. (Currently Amended) The method of claim 1 wherein upon receiving an event, the method further comprises:

assigning an execution context that is in idle state to process the event ~~a packet~~.

9. (Currently Amended) The method of claim 1 wherein binding an execution further comprises:

removing an event from the events for the event context in the event queue;

determining the event context; and

determining if the event context to which the event ~~a packet~~ belongs is already bound to an execution context.

10. (Currently Amended) The method of claim 9 wherein if the event context is already bound, binding an execution further comprises

placing the event ~~packet~~ in the event queue of the other execution context to which the event context associated with the event ~~packet~~ is already bound to;

unbinding the event context; and

returning to an idle state.

11. (Original) The method of claim 10 wherein if the event context is not already bound, binding an execution further comprises:

binding the execution context to that event context by updating a state of the execution context from idle to bound, updating the state of the event context from "not bound" to bound, and recording that this execution context is bound to this event context; and

processing the event.

12. (Previously Presented) The method of claim 11 wherein when the execution context completes processing the event, the execution context transitions to an unbinding state.

13. (Currently Amended) The method of claim 12 wherein when the execution context completes processing the event, the execution context checks its event queue for additional events to process.[[;]]

14. (Currently Amended) The method of claim 12 wherein if there is at least one event in the event queue, the execution context returns to the bound state, removes the event ~~packet~~ from the event queue and processes the event ~~packet~~, otherwise the execution context unbinds itself from the event context, and transitions to an idle state.

15. (Original) The method of claim 1 wherein the events are packets.

16-26. (Canceled)

27. (Currently Amended) A computer program product residing on a computer readable medium for dynamically binding an event context to an execution context in response to receiving events comprising instructions for causing a processor to:

maintain the execution context in an idle state until an event arrives at a head of a global event queue that is accessible by event contexts;

store events into [[a]] the global event queue ~~that is accessible by event contexts~~;

store events from the global event queue in per-execution context event queues; and

associate a FIFO event queue with the execution context to temporarily store events for the event context for a duration of the dynamic binding.

28. (Canceled)

29. (Currently Amended) The computer program product of claim 27 wherein upon receiving an event, the method further comprises instructions to:

assign an execution context that is in idle state to process the event ~~a packet~~.

30. (Currently Amended) The computer program product of claim 27 wherein instructions to bind an execution further comprises instructions to:

remove an event from the events for the event context in the event queue;

determine a event context; and

determine if the event context to which the event ~~a packet~~ belongs is already bound to an execution context.

31. (Currently Amended) The computer program product of claim 27 wherein if the event context is already bound, instructions to bind an execution further comprises instructions to:

place the event ~~a packet~~ in a FIFO event queue of the other execution context to which the event context associated with the event ~~packet~~ is already bound to;

unbind the event context; and

return to an idle state.

32. (Previously Presented) The computer program product of claim 27 wherein if the event context is not already bound, instructions to bind an execution further comprises instructions to:

bind the execution context to that event context by updating a state of the execution context from idle to bound;

update the state of the event context from "not bound" to bound;

record that this execution context is bound to this event context; and

process an event.

33-41. (Canceled)

42. (Currently Amended) A computer system comprising:

a processor including multiple processing engines, each processing engine including multiple event contexts;

circuitry to dynamically bind an event context to an execution context in response to receiving an event and maintain the execution context in an idle state until an event arrives at a head of a global event queue that is accessible by all event contexts;

[[a]] the global event queue ~~that is accessible by all event contexts~~ to store arriving events;

per-execution context event queues to store events from the global event queue; and

a FIFO event queue to temporarily store events for that event context for a duration of the binding.

43. (Previously Presented) The apparatus of claim 42 wherein the global event queue is used to queue events when the events first arrive into the computer system.

44-51. (Canceled)